**Analogy with Airy's linear theory.** *The author did not provide any physical justification of the analogy with Airy's wave theory: a linearised description of the propagation of gravity waves on the surface of a homogeneous fluid layer. A priori, this problem fundamentally differs from the SIA description of glacier ice tackled by MinSIA. In my view, there are several flaws:*

- *Airy's exponential decay is a consequence of water wave kinematics (Dingemans, 1997), not directly transferable to ice sheet mechanics. How is the horizontal smoothing in ice surface justified by the exponential decay along the vertical dimension in Airy's linear theory? Why larger ice thicknesses imply a larger length scale of smoothing?*
- *Ice behaves as a non-newtonian fluid, not considered in Airy's theory. Viscous dissipation in ice dampens short-wavelength perturbations, altering how surface effects propagate.*

*In summary, the analogy lacks physical justification. Ice thickness does not play the same role as wavelength in water waves decay, unless otherwise shown by the author. With the current description, the smoothing appears as an heuristic fix (as stated by the author: "the thickness-weighted average avoids artifacts at the boundaries of ice-covered area"), not a physics-derived approach. I encourage the author to either provide a detailed physical justification of the analogy or remove the reference to water wave propagation, thus re-framing the motivation.*

I did not claim that the smoothing approach can be formally derived from Airy's theory. It was only used as an example that short-wavelength oscillations do not accelerate the entire column of material as assumed in the shallow water/ice approximation, but typically have a limited depth of penetration. The depth of penetration is proportional to the wavelength only for linear problems (Airy's theory, but also for elastic materials). Qualitatively, however, it should also remain valid for nonlinear flow laws.

**Inadequacy of SIA.** *As described by Greve and Blatter (2004), the SIA is justified for large ice sheets where conditions generally vary little over horizontal distances (i.e., 5-10 times the local ice thickness). However, this is not often the case in mountain glaciers, where longitudinal and transverse coupling of stresses are important and should therefore be considered. To illustrate this inadequacy, Fig. 7.4 (Greve and Blatter, 2004) shows a scatter plot comparison of the First Order Approximation and the SIA velocities for Haut Glacier d'Arolla: the latter underestimates the velocities for values below 10 m/yr, and vice versa for velocities above this threshold.*

*Given that "The purpose of this paper is to challenge the hypothesis that the potential for further improvements in computational efficiency with classical numerical methods is limited (Jouvet et al., 2021)", I encourage the author to choose a more suitable stress balance approximation and discuss the differences with a more sophisticated description. If the SIA is nonetheless correct, then a simple plot as Fig 7.4 in Greve and Blatter (2004) will suffice to justify and discuss the choice over the domain. Note that the work of Jouvet et al, (2021) employed PISM output to train the emulator, capturing the physics contained in longitudinal stresses present in the SSA. It would be also convenient to discuss how the second-order SIA performs in this context (e.g., Ahlkrona et al, 2013.)*

It is not my intention to convince anyone to use the shallow ice approximation. The approach can probably be extended by lateral stresses. The simplest version would treat the respective derivatives of the velocity as an explicit component in the semi-implicit scheme. However, I just propose an efficient numerical scheme for the simplest glacial flow law.

**Lack of smooth-free reference.** *As stated by the author: "It would be desirable to use a simulation without smoothing as a reference scenario". There is an inevitably trade-off between smoothness and numerical stability. I consider that it is thus mandatory to have a reference simulation to quantify the deviation from the smooth-free solution. As stated in the paper: "the smoothing factor was set to $f = 0.25$ as a reference". Why is it so? This number seems arbitrary without further justification. In fact, it corresponds to a strongly oscillating result (Fig. 4) that need a nearly 108 m3 of ice in order to keep the ice surface consistent (Fig. 5). Please, provide detailed physical justification on why $f = 0.25$ should be a reference value.*

To be honest, I do not understand how you obtained the "strongly oscillating result" from Fig. 4 and "need a nearly $10^8$ m$^3$ of ice in order to keep the ice surface consistent" from Fig. 5. For $f = 0.25$ and $dt = 1/64$ yr (the reference), the 95 % oscillation (exceeded in 5 % of all time steps) is about $2 \times 10^{-5}$ m and the absolute maximum oscillation about $5 \times 10^{-4}$ m. Given all the uncertainties in glacier modeling, I would not consider oscillations of less than 1 millimeter strong. The added volume is always less than $3.5 \times 10^{-5}$ m$^3$ in each time step and less than 5 m$^3$ added over the entire 3000 yr simulation, which seems to be small in relation to the total ice volume of about $5 \times 10^{11}$ m$^3$. I tried to explain why simulations with smaller smoothing parameters are too expensive in lines 180-82.
Anyway, I just performed some additional tests. Without smoothing, I need $\delta t = 1/4096$ yr. Starting from $t = 2900$ yr (from the beginning, it would really be too expensive), the simulation takes 3 hours per year. The root mean square deviation in thickness between $f = 0.25$ and no smoothing is about 1.5 cm and the maximum deviation (over all points) about 35 cm. So these deviations are also completely irrelevant compared to all other uncertainties.

**Python implementation.** *As stated by the Editor, MATLAB is not the most "open software", so it would be convenient to provide the Python software under development and discuss the performance in the manuscript. In this line, NVIDIA recently announced the cuNumeric library, a drop-in replacement for the NumPy library that allows to run on multi-core CPUs, single or multi-GPU nodes, and even multi-node clusters without changing your Python code. Operations are executed by Legate's task engine and accelerated on one or many NVIDIA GPUs (if no GPU is present, on all CPU cores). Linking with the previous comment, as the author justify the absence of smooth-free reference simulations as a results of prohibiting computational costs, this apparent issue could be potentially overcome by using cuNumeric library. Either way, since the main focus of the paper is to show that there is still room for improvement in computational efficiency with classical numerical methods, I encourage the author to include a section where parallel performance is elaborated.*

Looks like a good suggestion, although I do not know how deep the cuNumeric library goes into solving linear equation systems.
Concerning the smooth-free reference solution, however, I have only one (even quite old) NVIDIA card available. And if I want to use a smooth-free reference scenario with $\delta t = 1/4096$ yr, I would need simulations with different smoothing factors in order to separate the effect of smoothing from the effect of $\delta t$. Then it would become expensive again and it would still be much effort for deviations of a few centimeters.
Anyway, SciPy seems to be not as good as I expected concerning linear solvers. So I am still struggling with the Python implementation and far off from reaching the same efficiency as the MATLAB version.

**Missing comparison with measured glacier velocities.** *I consider that the perturbation induced by smoothing should be rather framed in the context of observed glacier velocities. It is expected that timestep and spatial resolution will impact the presence of numerical artefacts such as the oscillations discussed, but a comparison with observed velocities is fundamental. If this paper aims at keeping "classical numerics competitive", it requires some sort of validation with measured velocities. My suggestion would be to compare MinSIA results with observed values as Jouvet et al. (2021). This will serve as a validation test to quantify to what extent smoothing perturbs the velocity field.*

Again, it is not my aim to justify the shallow ice approximation by tuning the parameters to reproduce any measured velocities. Without smoothing, there are considerable oscillations at the surface due to the staircase-like ice surface, and these oscillations vanish with increasing smoothing. Nothing beyond.

**Overstated stability claim.** *Line 107 of the manuscript reads: "This semi-implicit scheme already ensures stability for arbitrary time increments $\delta t$". This claim is quite strong and generally incorrect for this specific scheme. If $D$ changes rapidly in space or time, evaluating it explicitly can lead to instabilities for large $\delta t$. The scheme is not unconditionally stable. In fact, the author later states in the paper that special focus is needed on the timestep to avoid numerical oscillations (see my next comment) and even discusses situations where the CFL criterion is relevant. Please, revise other vague statements regarding numerical stability.*

For this discussion, it would be helpful to know which definition of instability you are referring to. I refer to the definition that the spatial variation in the solution increases with decreasing $\delta x$ at fixed $\delta t$. For this scheme, the oscillation is almost independent of $\delta x$ at fixed $\delta t$. This means that Fig. 4 would look basically the same at larger $\delta x$. The oscillation does also not increase strongly through time – definitely not exponentially as it is typically the case for unstable schemes. The increase in oscillation is even weaker than the increase in maximum thickness. So I would say it is formally unconditionally stable, but with a poor accuracy if smoothing is too weak in relation to $\delta x$ and $\delta t$. So please tell me which definition of instability you are referring to since I cannot "revise other vague statements regarding numerical stability" otherwise.

**Focus on "time increment" $\delta t$.** *The paper reads that: "[dots] the systematic error in the volumetric balance is negligible compared to the immediate effect of the staircase oscillations on accuracy. So focus should be on limiting $\delta t$ to avoid these oscillations". Can we simply conclude that, given the numerical nature of a semi-implicit scheme, the stability is determined by the timestep? (and therefore contradicts the previous claim that the semi-implicit scheme ensures stability for arbitrary timesteps). If so, a fully implicit scheme would overcome this issue? Further experiments are needed to support or reject this hypothesis.*

Also not sure whether I got your point correctly. Yes, there is a sharp transition between "ok" and oscillations leading to a poor accuracy (instability according to your definition), and this transition can be quantified in terms of a maximum $\delta t$ depending on $\delta x$ etc. A fully implicit scheme that overcomes this limitation should presumably be a "genuine" implicit scheme and not an iterative scheme built around the mixed scheme. Bueler (2016, doi 10.1017/jog.2015.3) proposed such a scheme, admitting later (Bueler, 2023, doi 10.1017/jog.2022.113) that "The scaling of the Bueler (2016) implicit SIA solver is not enough better than such an explicit scheme, however." This finding is consistent with my findings on the semi-implicit scheme. At $f = 1$ and $\delta t = 0.25$ yr, the highest Fourier number is $> 1500$. As stated above, we have to decrease $\delta t$ to 1/4096 yr without smoothing, which reduces the Fourier number to about 1.5. This is still higher than for an explicit scheme, but practically not enough to justify solving a big linear equation system in each step. The same holds for the predictor-corrector schemes you are promoting in the next point. So please let me know what kind of further experiments you would like to see.

**Timestepping and redundant sections.** *I would suggest merging Section 4.2 (The maximum time increment) and 5 (Finding the best time increment) in a single section. All numerical results regarding timestepping should be described and discussed therein.*
*Moreover, I find a great amount of effort on the present work while lacking some important points. For instance, Cheng et al. (2017) introduced an adaptive time step control for simulations of the evolution of ice sheets using Elmer/Ice (Gagliardini et al., 2013). Semi-implicit and fully implicit methods are compared for a number of discretization stencils. I consider that if the problem of "finding the best time increment" is to be tackled, the paper should dive into the predictor-corrector (among others) approaches and showcase the performance in MinSIA.*

The first draft of the manuscript contained such a merged section, but then I decided to separate the practical aspects of finding a "good" $\delta t$ from the numerical analysis (and included some results from the Alpine examples). I still think that it is better to keep this more practical section for those who do not want to go though the entire numerical analysis.
And of course, it would be possible to built an adaptive time step control around the step() method, based on the maximum oscillation. However, I want to keep the implementation is small as possible at the moment.
Furthermore, I have no idea what to write about the predictor-corrector schemes in the reference Cheng et al. (2017). As far as I can see, these schemes do not increase the maximum achievable time increment, but only yield a higher accuracy than an explicit scheme for small $\delta t$.

*Figure 1. This plot is hard to interpret: colour bar is missing, legend is missing, spatial scale is missing, inset with geographical zoom-out is also missing. Please, improve the figure so that they are as much self-explicative as possible.*

I can try.

*It would be very convenient to include explicit discretization schemes implemented in MinSIA. Section 3 (Numerical scheme and implementation) elaborates on the smoothing algorithm, but the finite volume and upstream diffusivity schemes are not given in the manuscript. An appendix with the explicit discretization is beneficial for reproducibility and future comparison.*

I can do this, but I feel that it would be a bit trivial. Finite-volume diffusion on a regular grid should be clear for those who get so deep into the implementation. And then taking the diffusivity from the node with the higher surface elevation is also simpler in words than in mathematical symbols.

*I wonder how the CFL criterion looks like overlaid in Fig. 7. It is illustrative to show the timestep restriction imposed by the CFL criterion for different resolutions and smoothing values. Moreover, Fig. 7 should also include the deviation from the smooth-free reference simulation. Large values of the smoothing factor could imply unrealistic velocity fields.*

Since the velocities are almost independent of the smoothing factor (as long as the oscillations are not too strong), the CFL criterion would be a horizontal line. Roughly $\delta t = 1/20$ year for $\delta x = 25$ m, $\delta t = 1/10$ year for $\delta x = 50$ m, etc. Not sure whether this information provides new insights. And what do you mean with "deviation from the smooth-free reference simulation"? I could include a data point left down for with $f = 0$ (unfortunately, a bit difficult with the logarithmic scale), $\delta t = 1/4096$ yr and the respective values for the other $\delta x$, but I am not sure whether it improves the figure. And again, large values of the smoothing factor have a limited effect on the velocity field and do not imply unrealistic velocity fields.

*Line 5: What does this mean: "MinSIA is designed for simulations with several million nodes on standard desktop PCs"? What does the author mean by node here? Regular desktop PCs usually have only $\sim$ 16-32 CPU cores.*

Nodes of the finite-volume grid.

*Lines 264-270. This paragraph could be synthesized by plotting the computing time as a function of different parameters (e.g., $\delta t$, $\delta x$, $f$, etc.). In log-scale, the slope of the linear fit will show the exponential dependency.*

I thought about this, but I have a very limited amount of timing data available. Since the computations were performed on different PCs with different numbers of processes running, I cannot reconstruct the computing times from the metadata of the output files.